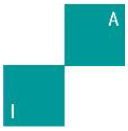
Inteligencia Artificial 46(2010), 34-44
doi: 10.4114/ia.v14i46.1520



INTELIGENCIA ARTIFICIAL

http://erevista.aepia.org/

Comparison of Recombination Operators in Panmictic and Cellular GAs to Solve a Vehicle Routing Problem

Carlos Bermudez[1], Patricia Graglia[1], Natalia Stark[2], Carolina Salto[3] and Hugo Alfonso[3]
Universidad Nacional de La Pampa
General Pico, Argentina
[1]{bermudezc,pmg_xxi}@yahoo.com [2]nataliastark@gmail.com [3]{saltoc,alfonsoh}@ing.unlpam.edu.ar

Abstract The Vehicle Routing Problem (VRP) deals with the assignment of a set of transportation orders to a fleet of vehicles and the sequencing of stops for each vehicle to minimize transportation costs. This paper presents two different genetic algorithm models (panmitic and cellular models) for providing solutions for the Capacitated VRP (CVRP), which is mainly characterized by using vehicles of the same capacity. We propose a new problem dependent recombination operator, called Best Route Better Adjustment recombination (BRBAX), which incorporates problem specific knowledge such as information about the routes constitution. A comparison of its performance is carried out with respect to classical recombination operators for permutations. A complete study of the influence of the recombination operators on the genetic search is presented. The results show that the use of our specialized BRBAX operator outperforms the others more generic operators for all problem instances under all metrics. The deviation between our best solution and the best-known one is very low, under 0.91%.

Keywords: Capacitated Vehicle Routing Problem, Recombination, Genetic Algorithms, Cellular Genetic Algorithms.

1 Introduction

The Vehicle Routing Problem (VRP) [12] consists in delivering goods to a set of customers with known demands through minimum cost vehicle routes, beginning and finishing at the depot. The VRP is a NP-hard problem [19] and has many industrial applications, being studied both theorical and practically. There are a large number of extensions to the canonical VRP [28]. One basic extension is known as the Capacitated VRP (CVRP), the one we focus on in this paper. The CVRP can be defined as follows. Let $G = (V, E)$ be a complete undirected graph consisting of $c + 1$ nodes (V), and a set of edges E with non-negative weights and with associated travel times. The nodes represent c customers and the additional node is designated as the depot. Each customer i has associated a demand q_i. There are k identical vehicles of capacity Q, which must deliver goods to a set of customers. Each route must start and end at the depot, and each customer must be served by exactly once by one vehicle. The problem is to minimize the total travel distance of a routing plan such that the total demand of any route does not exceed a vehicle capacity Q (the capacity constraint) and the duration of any route does not exceed an upper limit L (the route duration limit). Note that the above described VRP with the route duration limit is often separated from CVRP and called Distance-constrained VRP (DVRP).

Due to the practical relevance of VRP and its NPhardness, many heuristic or metaheuristic solution methods have been proposed to solve the VRP. Some examples include Tabu Search [10], Simulated Annealing [24], Ant Colony [7], Evolutionary Algorithm [6, 30], among others. In this work, we have used Evolutionary Algorithms (EAs) [5, 22], in particular Genetic Algorithms (GAs), to find a minimum total travel distance of a route plan sat-

ISSN: 1988-3064(on-line)

isfying the vehicle capacity constraint. In recent years, GAs have drawn a great deal of attention from researchers to solve the CVRP due to its robustness and flexibility [8, 23].

GAs deal with a population of tentative solutions, each one encodes a problem solution on which genetic operators are applied in an iterative manner to progressively compute new higher quality solutions. Usually, most GAs use a single population (panmixia) of individuals where an individual may mate with any other (see Figure 1). In contrast, there also exists some tradition in using structured GAs (where the population is decentralized in some way), especially in relation to their parallel implementation. Cellular GAs are a subclass of GAs in which the population is structured in a specified topology (usually a toroidal mesh of dimensions $d = 1, 2$, or 3). In a cGA, the genetic operations may only take place in a small neighborhood of each individual (see Figure 1). The pursued effect is to improve on the diversity and exploration capabilities of the algorithm (due to the presence of overlapped small neighborhoods) while still admitting an easy combination with local search to improve on exploitation at the level of each individual. Research in cGAs is a healthy field with recent advances in theory [1, 15, 16] and new results reporting their usefulness in maintaining diversity and promoting slow diffusion of solutions through the population grid [26].

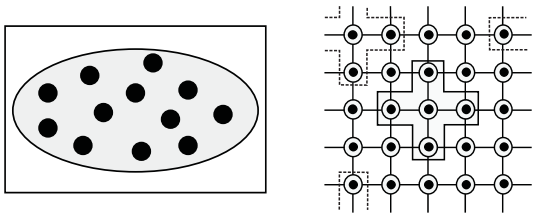


Figure 1: Panmictic and cellular populations.

We here investigate the advantages of using a special built-in recombination in GAs to solve CVRP, that incorporate problem specific knowledge, such as information about the customer's demand and the distance of each route. The objective of this work is to find an effective recombination operator and to quantify the effects of including it into the algorithm procedure. Furthermore, we perform an empirical study where we compare the performance of the new recombination operator (BRBAX) with other classical recombination operators used in the literature to solve this problem. Also we use different mutation operators in order to determine the best combination of genetic operators. The central idea here is to evaluate the GA and cGA performance including these operators. Furthermore we test the performance to apply or not local search techniques, which was probed to improve the algorithm performance [2].

The remainder of this paper is organized as follows. The proposed GA and cGA are described in Section 2 and 3, respectively. In Section 4 we review the representation and the fitness function used for CVRP. In Section 5 we described studied genetic operators and present a detailed description of the new recombination operator proposed. Details of implementation are summarized in Section 6 and in Section 7 and 8 we reports on the algorithm performances. Finally, in Section 9 we give some conclusions and analyze future lines of research.

2 Genetic Algorithm for CVRP

In this section we present a simple GA for solving the CVRP. In Algorithm 1 we can see the structure of a basic generational GA in which we will now explain the steps for solving our routing tasks. By simulating evolution, this algorithm maintains a population ($P(t)$) of multiple individuals, which evolve throughout multiple generations by allowing the reproduction and further enhancement of the fittest ones.

Algorithm 1 Pseudocode of a genetic algorithm.

```
1: t = 0; {current generation}
2: Initialize(P(t));
3: Evaluate(P(t));
4: while not max_generations do
5:     P'(t) ←Evolve(P(t)); {recombination and mutation}
6:     Evaluate(P'(t));
7:     P(t + 1) ← Select_new_population(P'(t) ⋃ P(t))
8:     t = t + 1;
9: end while
10: Show_best_solution();
```

This algorithm creates an initial population (initialize($P(t)$)) of μ solutions in a random (uniform) way, and then evaluates these solutions. After that, the population goes into a cycle where it undertakes evolution. This consists of a recombination-mutation-selection cycle to compute improved individuals until a maximum number of generations (*max_generations*) is reached. The best solution is identified as the best individual ever found, which minimizes the route lengths and respects the capacity constraint.

In fact, this metaheuristic provides not only one solution to the problem, but a set of solutions of good quality when the search finishes: the final population is made up of a well adapted individuals containing different suboptimal solutions for the problem at hands. GAs are guided by the values computed by an objective function for each tentative solution until an optimum or an acceptable solution is found.

3 Cellular Genetic Algorithms for CVRP

In this section we present the structure of a hybrid cGA proposed to solve CVRP. In a cGA the concept of *neighborhood* is introduced; this means that individuals only interact with theirs neighbors in the breeding loop (see Figure 1). The most common population topology used in cGA is a 2-D toroidal grid holding all the individuals, which is adopted in this work. The existence of small overlapped neighborhoods help in exploring the search space because the induced slow diffusion of solutions through the population provides a kind of exploration, while exploitation takes place inside each neighborhood by genetic operators.

In Algorithm 2 we show the pseudocode of the cGA which is high-level description of the algorithm for a 2-D static grid of size `HEIGHT`×`WIDTH` and for formally simultaneous update of all the cells. As it can be seen, a cGA starts with the cells (individuals) in a random state and proceeds by successively updating them using evolutionary operators, until a termination condition is met. Updating a cell in a cGA means selecting two parents in the individual's neighborhood (including the individual itself), applying the genetic operators to them, and improving the individual by local search procedure consisting in applying *2-Opt*[11] and *1-interchange*[24], which are well-know local optimization methods. Finally, a replacement of the individual is carried out following a given replacement policy in a new auxiliary population. After applying the reproductive cycle to all the individuals, the current population is replaced by the auxiliary one for the next generation.

Algorithm 2 Pseudocode of a synchronous cGA.

```
1: while not Stop_condition() do
2:    for x ← 1 to WIDTH do
3:       for y ← 1to HEIGHT do
4:          n_list←Get_neighborhood(position(x,y));
5:          parents←Local_select(n_list);
6:          aux_indiv←Evolve(parents); {recombination and mutation}
7:          aux_indiv←Local_search(aux_indiv);
8:          Evaluate(aux_indiv);
9:          Insert_if_better(position(x,y),aux_indiv,aux_pop);
10:      end for
11:   end for
12:   cga.pop←aux_pop;
13: end while
14: end_proc Steps_Up;
```

4 Representation and Fitness Function for CVRP

In genetic algorithms, individuals represent candidate solutions. A candidate solution to an instance of the CVRP must specify the number of vehicles required, the allocation of the demands to all these vehicles, and also the delivery order of each route. The adopted representation consists in a permutation of integer numbers (following the Alba and Dorronsoro' ideas [2]). Each permutation will contain both customers and route splitters, which delimits different routes. The permutation of numbers $[1 \dots n]$ will have a length of $n = c + k - 1$ for representing a solution for the CVRP with c customers and $k - 1$ route splitters. Each route is composed of the customers between two route splitters in the individual. Customers are represented with numbers $[1 \dots c]$, while the $k - 1$ route splitters belong to the range $[c + 1 \dots n]$.

Note that due to the nature of the chromosome (permutation of integer numbers) route splitters must be different numbers, although it should be possible to use the same number for designating route splitters in the case of using other possible chromosome configuration. The number of vehicles k is calculated as follows:

$$k = total\_demand\_routes/vehicle\_capacity \times 1.3 \quad (1)$$

In this study, the length of routes is minimized independently of the number of vehicles used. Empty routes are allowed in this representation simply by placing two route splitters contiguously without customers between them.

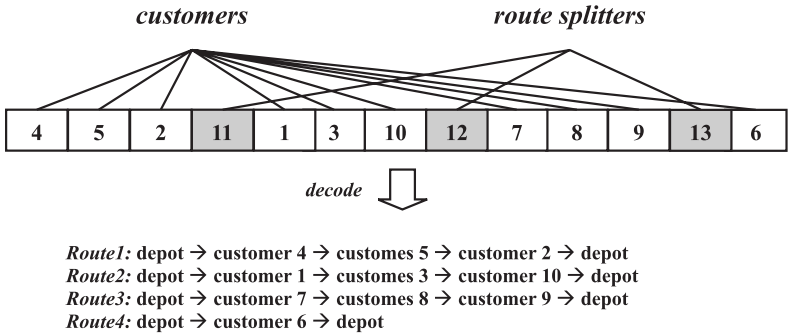


Figure 2: Individual representing a solution for 10 customers and 4 vehicles.

For example, in Figure 2, we plot an individual representing a possible solution for a hypothetical CVRP instance with 10 customers using at most 4 vehicles. Values $[1, \ldots, 10]$ represent the customers while $[11, \ldots, 13]$ are the route splitters. Route 1 begins at the depot, visits customers 4, 5, 2 (in that order), and returns to the depot. Route 2 goes from the depot to customers 1, 3, 10 and returns, and so on.

The fitness value $f(S)$ assigned to every individual S is computed as shown in Equation 2, as defined in [18, 17].

$$f(S) = f_{RoutePlanCost}(S) + \lambda \times overcap(S) \quad (2)$$

Function $f(S)$ is computed by adding the total costs of all the routes ($f_{RoutePlanCost}(S)$), and penalizes the fitness value only in the case that the capacity of any vehicle is exceeded. The function $overcap(S)$ returns the overhead in capacity of the solution with respect to the maximum allowed value of each route. This value returned by $overcap(S)$ is weighted by multiplying them by factor λ. In this work we have used $\lambda = 1000$ [13].

5 Genetic Operators

In this section we review the studied recombination and mutation operators and present a detailed description of the new recombination operator devised.

5.1 Recombination Operators

We have studied three recombination operators, from which two have been proposed for permutations representations in the past and used in previous works solving the VRP [2, 21, 25, 27, 30]. Partial Mapped Crossover (PMX) [17] focuses on combining the order information from the two parents, taking into account the position and order of as many customers or splitter routes as possible. The other one, Edge Recombination Crossover (ERX) [29], focuses on the links between customers (edges) preserving the linkage between them. The main disadvantage of these traditional operators is that they do not incorporate knowledge of the problem to carry out the genetic exchange of information. The last operator, called Best Route Better Adjustment recombination (BRBAX), is a new one tailored for this problem. This operator transmits the best routes (groups of customers) of one parent to the offspring. We consider good routes as the ones which make the best use of the vehicle capacity and also minimize the total travel distance. The BRBAX operator works as follows (see Algorithm 3). In a first step, BRBAX sorts the k routes of $parent1$ in an increasing way regarding the difference between the demand of each route and the vehicle capacity. Then, it selects the best $k/2$ routes and placed them in the first positions of the *child*. The customers belonging to the selected routes are placed in the child separated by route splitters. Finally, the remaining positions of the child are filled with the customers or route splitters which do not belong to the inherited routes, in the order they appear in the other parent, $parent2$. Figure 3 gives an example for the route transfers in the course of a recombination operation and also the filling process of the remaining positions.

5.2 Mutation Operators

The mutation operators will play an important role during the evolution since it is in charge of introducing a considerable degree of diversity in each generation. We have tested three mutation operators, they are *Insertion*

Algorithm 3 Pseudocode of a BRBAX.

```
1: proc Best_Routes(parent_1,parent_2);
2: i_1 ← parent_1;
3: i_2 ← parent_2;
4: Order_by_aptitud(i_1, i_2);{i_1 is the best parent i_1}
5: Routes ← (k/2);
6: l ← List_of_route(i_1);
7: {Complete the child with the best route of best parent}
8: for j ← 1 to Routes do
9:    rn ← Select_best_route(l);
10:   child ← Copy_route(rn);
11:   Delete_route_of_list(rn, l);
12: end for
13: {Complete the child with the customer or route splitter of i_2}
14: for i ← last_position to n do
15:   aux ← First_element_not_inherited(child, i_2);
16:   child ←Copy_element(aux, i);
17: end for
18: return child;
19: end proc
```

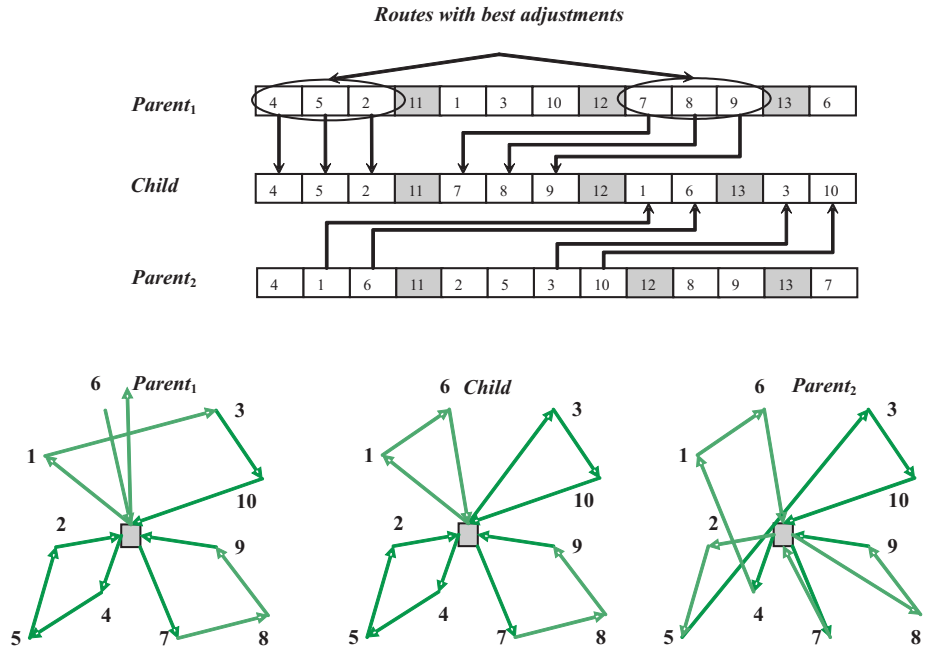


Figure 3: Example of BRBAX recombination.

[14], *Swap* [5], and the last one, called *Combined*, which is a combination of the previous ones [2]. The first two mutation operators are well-known methods found in the literature, and typically applied in routing problems. The *Insertion* operator selects a gene (either customer or route splitter) and inserts it in another randomly selected place of the same individual. On the other hand, *Swap* consists in randomly selecting two genes in a solution and exchanging them. Note that the induced changes might occur in an intra or inter-route way in all the two operators. The last operator, called *combined*, consists in applying *Insertion* and *Swap* operations to each individual with equal probability.

6 Implementation

Now we will comment on the actual implementation of the algorithms to ensure that this work is replicable in the future. As we said from the beginning, a GA will be evaluated for a set of selected instances of the CVRP, in order to better know on their algorithmic effectiveness. The GA approach will be tested with all possible combinations of recombination (BRBAX, ERX, and PMX) and mutation operators (*Insertion*, *Swap* and *Combined*). All these

algorithms have been compared in terms of the quality of their results. Furthermore, the GA is compared with a cGA in a numerical way, to show the numerical advantages of the Cellular Algorithms. The best settings for the problem instances of the CVRP are the following.

GA Parameters. The population size was set to 512 individuals. By default, the initial population is randomly generated. The maximum number of generations was fixed to 7500. The parents were selected using binary tournament. The recombination operators were applied with a probability of 0.65, while the mutation probability was set to 0.1. These parameters (population size, stop criterium, probabilities, etc.) were chosen after an examination of some values previously tested. The algorithms were implemented in C++ using the MALLBA software package [3] and executed on an Intel Pentium 4 at 2.4 GHz with 4 GB, under SuSE Linux with 2.4.19 kernel version.

cGA Parameters. The population size was set to 100 individuals disposed in a square 2D toroidal grid. The population was evolved by 300 cycles. The initial population was randomly generated. The parents were selected using binary tournament between the individuals belonging to the neighboorhood (L5 model). The recombination operators were applied with a probability of 0.65, while the mutation probability was set to 0.85. Local optimization was applied to every individual after recombination and mutation operators, this step consists in applying 2-Opt and 1-Interchange to the same individual for 20 movements and returning the best solution between the best ones found by 2-Opt and 1-Interchange, or the current one if it is better. These parameters were chosen after an examination of some values previously tested. The algorithms were implemented in Java using the JCell software package [20] developed by the NEO group at Malaga University. The execution platform was an AMD Phenom 8450 Tripe-core Processor at 2.4 GHz with 4 GB, SLACKWARE 12 with 2.6.27 kernel version.

The computational tests were carried out with the standard CVRP benchmarks of Christofides, Mingozzi and Toth [9]. This set of data files is composed of fourteen instances problems consisting of $50 - 200$ customers (nXX) and using 5 to 18 vehicles (kXX). For example, instance named CMT-n51-k5 consists of 50 customers plus the depot and has 5 vehicles. Half of the instances, denominated CMT-nXX-kXX, are basic VRP problems that do not include route length restrictions. The remaining seven instances, called CMTd-nXX-kXX, has the same customer locations as the previous seven, but they are subject to additional constraints like limited distances on routes and drop times for visiting each customer. Instances CMT-n101-k10, CMT-n121-k7, CMTd-n101-k11, and CMTd-n121-k11 are clustered ones meanwhile the eight remaining instances has the customers located randomly in the plane.

7 Computational Analysis of GA

In this section we will summarize the results of applying the proposed algorithms on all the problem instances. For each algorithm variant we performed 30 independent runs per instance using the parameter values described in the previous section. In order to obtain meaningful conclusions, we have performed an analysis of variance of the results. When the results followed a normal distribution, we used the t-test for the two-group case, and the ANOVA test to compare differences among three or more groups (multiple comparison test). We have considered a level of significance of $\alpha = 0.05$, in order to indicate a 95% confidence level in the results. When the results did not follow a normal distribution, we used the non-parametric Kruskal Wallis test (multiple comparison test), to distinguish meaningful differences among the means of the results for each algorithm.

Table 1 shows the comparative results on the benchmarks of Christofides et al. The figures in this table stand for the best obtained fitness values (column *Best*) and the average objective values of the best found feasible solutions (column *Avg*). The minimum best values are printed in bold.

These results clearly show that the GA using *Insertion* operator outperforms the GAs using any other mutation operator in terms of solution quality, for the majority of the instances. *Combined* mutation reaches the best values for CMT-n200-k17, CMT-n121-k7 and CMT-n101-k10 with PMX recombination and for CMT-n101-k8 and CMT-n101-k10 with ERX. Using the test of multiple comparisons, we have verified that the mentioned differences among the results are statistically significant.

Regarding the average number of generations to reach the best value, *Swap* reaches generally their best solutions in a less number of generations but these solutions are of poor quality, as indicated in the previous paragraph. The exceptions are the instances CMT-n200-k17, CMT-n121-k7 and CMT-n101-k10, for which the PMX operator is the fastest one. Moreover, *Insertion* and *Combined* mutations do not present mean significant differences, as shown by the multiple comparison tests performed in each case.

As a preliminary conclusion, we can indicate that *Insertion* is the most suitable mutation of those studied, regarding solution quality and also minimum effort to reach the best value.

Now we turn to analyze the results obtained for the different recombination operators with the GA using *Insertion* (see **Insertion** columns of Table 1). For all instances, BRBAX reaches the best solutions but needs larger number of generations than the rest. This operator also gets the best averages values (statistically corroborated),

Table 1: Experimental results for the GA with all recombination and mutation operators.

Inst	Cross	Insertion		Swap		Combined	
		best	*avg*	*best*	*avg*	*best*	*avg*
CMT-n76-k10	BRBAX	**906.48**	1056.60	1962.81	2172.97	965.96	1105.97
	ERX	1086.70	1408.60	1892.01	2119.20	1195.93	1448.91
	PMX	921.74	1028.30	1822.45	2060.71	963.90	1082.29
CMT-n101-k8	BRBAX	**945.14**	1094.80	2361.8	2645.92	1096.28	1229.36
	ERX	1142.75	1399.60	2190.21	2440.55	1075.72	1441.81
	PMX	977.78	1082.10	2205.57	2480.12	1065.61	1167.46
CMT-n151-k12	BRBAX	**1377.53**	1507.60	3951.34	4284.78	1624.47	1748.28
	ERX	1618.78	1985.50	3011.62	3545.34	1690.40	2084.15
	PMX	1441.90	1563.70	3617.44	4046.62	1523.14	1716.41
CMT-n200-k17	BRBAX	**1964.08**	2109.20	5715.06	6787.96	2284.22	2509.79
	ERX	2370.55	2919.40	4277.68	5600.36	2416.88	3054.95
	PMX	6495.71	47731.00	5232.69	5852.00	2160.18	2381.08
CMT-n121-k7	BRBAX	**1737.77**	2050.80	4762.94	5292.54	2103.54	2274.52
	ERX	2279.79	2568.50	4283.55	5439.12	2477.22	2806.27
	PMX	5652.00	14963.00	4417.25	5061.18	2060.03	2336.40
CMT-n101-k10	BRBAX	**1062.66**	1190.90	2809.05	3138.07	1183.66	1334.32
	ERX	1235.73	1611.20	2284.26	2768.97	1135.00	1500.90
	PMX	3165.65	5858.70	2311.12	3049.54	1147.51	1299.50

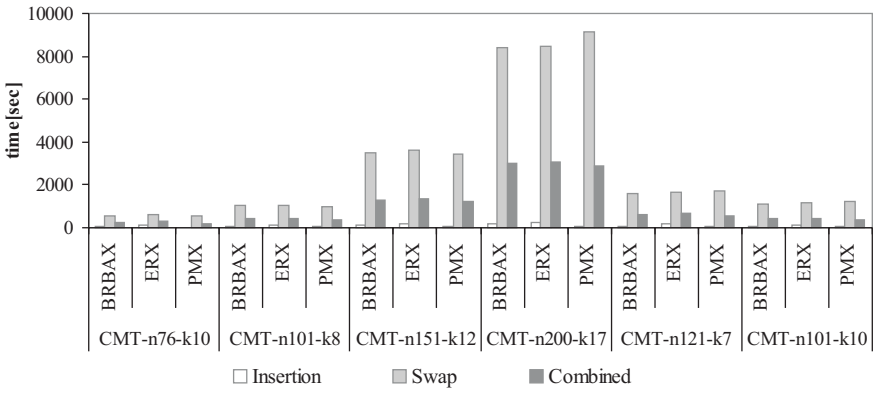


Figure 4: Mean spent time in the total search for GA variants.

except for CMT-n76-k10 and CMT-n101-k8 instances. For these last two instances, PMX reaches the best mean values, but the differences with BRBAX are not significant, as it is shown by the results of the multiple comparison tests.

The BRBAX outperforming is due to the fact that, in some way, BRBAX exploits the idea behind building blocks, in this case defined as a route, i.e. a group of customers and tends to conserve good routes in the offspring produced during recombination. As a result, BILX can discover and favor compact versions of useful building blocks.

With respect to the time spent in the search for all the operators proposed in this study, Figure 4 shows that Insertion mutation is the fastest mutation for all instances. By the other hand, PMX is the fastest recombination operator because it does not need neither to select the best routes to transmit to the child nor to build the edges matrix as BRMAX or ERX, respectively. The ranking order is PMX, BRBAX and ERX from faster to slower ones.

8 Computational Analysis of cGA

In this section, we describe the results obtained for our proposed cGA (see Section 3) using BRBAX to solve the whole set of instances of CVRP (see . All the results provided here are the mean of 30 independent runs.

Table 2 summarizes the results obtained for the cGA+BRBAX. The most relevant aspects measured in this comparison are the following ones: the best known value for each instance (column *best known*), the best total travelled length of the solutions found in our runs (column *best found*), the deviation (in percentage) between our best solution and the previously best-known one (column *error*), the success rate (column *hits*) in percentage, the

Table 2: Experimental results for the cGA+BRBAX.

Inst	*best known*	*best found*	*error*	*hits(%)*	*avg*$\pm\sigma$	*eval*
CMT-n51-k5	524.61	**524.61**	0.00	84	525.22 $\pm$ 1.57	106700
CMT-n76-k10	835.26	**835.26**	0.00	7	840.56 $\pm$ 4.71	440995
CMT-n101-k8	826.14	**826.14**	0.00	8	830.68 $\pm$ 3.33	623956
CMT-n151-k12	1028.42	1029.87	0.14	0	1044.93 $\pm$ 4.46	1372206
CMT-n200-k17	1291.45	1303.2	0.91	0	1324.01 $\pm$ 7.11	2175806
CMTd-n51-k6	555.43	**555.43**	0.00	18	558.64 $\pm$ 2.19	131464
CMTd-n76-k11	909.68	**909.68**	0.00	34	912.84 $\pm$ 2.94	222566
CMTd-n101-k9	865.94	**865.94**	0.00	44	869.85 $\pm$ 4.82	402146
CMTd-n151-k14	1162.55	1164.12	0.14	0	1173.16 $\pm$ 7.86	1216652
CMTd-n200-k18	1395.85	1406.46	0.76	0	1422.05 $\pm$ 9.71	1965968
CMT-n121-k7	1042.11	1043.89	0.17	0	1135.53 $\pm$ 44.46	1426162
CMT-n101-k10	819.56	**819.56**	0.00	78	823.74 $\pm$ 7.96	219122
CMTd-n121-k11	1541.14	1544.22	0.20	0	1563.66 $\pm$ 14.04	842158
CMTd-n101-k11	866.37	**866.37**	0.00	88	867.19 $\pm$ 2.4	301860

Table 3: Experimental results for the GA and cGA

Inst	*GA*	*cGA*	*error*
CMT-n76-k10	906,48	**835,26**	71,22
CMT-n101-k8	945,14	**826,14**	119,00
CMT-n151-k12	1377,53	1029,87	347,66
CMT-n200-k17	1964,08	1303,20	660,88
CMT-n121-k7	1737,77	1043,89	693,88
CMT-n101-k10	1062,66	**819,56**	243,10

average values of the solutions found in all the independent runs along with their standard deviations (column *avg*$_{\pm\sigma}$) and the average number of evaluations made to find the best solution (column *eval*).

From Table 2 we can observe that cGA+BRBAX reaches the best known value in eight of the fourteen instances (*best* values are printed in bold). For the rest of the instances, the deviation is less than 1%, which emphasizes the good quality of results obtained. Higher errors are present for instances with 200 costumers. Regarding the percentage of times that the best values were obtained, we observe a variation from 7 to 88% . The instances with more than 120 costumers do not find the best-known solution. Furthermore, these instances are the ones with highest computational effort.

Now, we continuous with the comparative analysis of the behavior of our GA+BRBAX and cGA+BRBAX in order to show the improvement on the results obtained. This analysis is carried out with only six instances, because the behavior of the GA+BRBAX was analyzed with pure instances, i.e., the ones which does not incorporate route duration constraint. The results of this comparison are shown in Table 3. Figures in column *error* represent the difference between the fitness values of the best found solutions of each algorithm. As can be seen, the cGA+BRBAX is able to improve the solution qualities of all instances. It is important to indicate that cGA applies a local search procedure to improve the quality of each found solution. The algorithm cGA+BRBAX find the optimum for half of the tested instances. Summarizing this results, we can indicate that structuring the population has been useful for improving the results, since the population diversity is better maintained with regard to non-structured population.

In the following we proceed with the comparison of the behavior of our cGA+BRBAX with a cellular evolutionary algorithm (cGA) proposed by Alba and Dorronsoro [4]. The analysis is performed using the whole set of instances of Cristofides et al., although the work of [4] includes all the known benchmarks for the problem. We are comparing the same algorithm in the two cases with the difference being the recombination operator used: cGA apply ERX. Moreover, cGA+BRBAX uses the same configuration and parameterizations that cGA+ERX, in order to replicate the experimentation presented in [4].

Table 4 shows the best found values for cGA+ERX (extracted from [4]) and cGA+BRBAX and the difference between them (column *dif*). Both algorithms reach the same quality solutions in eight of the fourteen instances, values in column *dif* are equal to 0.00. The algorithm cGA+ERX reaches a best solution than cGA+BRBAX for two instances: CMT-n121-k7 and CMTd-n121-k11, both with 121 customers. In the rest of the instances, cGA+BRBAX obtains better solutions than cGA+ERX, although those solutions are different from the best-known value. The reported error for this instances are less than 1%.

Figure 5 shows the success rate for cGA+BRBAX and cGA+ERX. Our cGA obtains higher hit rates in six of the fourteen instances, meanwhile in instances CMT-n101-k10 and CMTd-n51-k6 the opposite situation is observed. For the rest of the instances the algorithms does not obtained the best known value in any of the runs.

Table 4: Experimental results for the cGA+ERX [4] and cGA+BRBAX.

Inst	best known	best found			error	
		cGA+ERX	cGA+BRBAX	dif	cGA+ERX	cGA+BRBAX
CMT-n51-k5	524.61	524.61	524.61	0.00	0.00	0.00
CMT-n76-k10	835.26	835.26	835.26	0.00	0.00	0.00
CMT-n101-k8	826.14	826.14	826.14	0.00	0.00	0.00
CMT-n101-k10	819.56	819.56	819.56	0.00	0.00	0.00
CMT-n121-k7	1042.11	1042.12	1043.89	-1.77	0.00	0.17
CMT-n151-k12	1028.42	1035.22	1029.87	5.35	0.66	0.14
CMT-n200-k17	1291.45	1315.67	1303.20	12.47	1.88	0.91
CMTd-n51-k6	555.43	555.43	555.43	0.00	0.00	0.00
CMTd-n76-k11	909.68	909.68	909.68	0.00	0.00	0.00
CMTd-n101-k9	865.94	865.94	865.94	0.00	0.00	0.00
CMTd-n101-k11	866.37	866.37	866.37	0.00	0.00	0.00
CMTd-n121-k11	1541.14	1543.63	1544.22	-0.59	0.16	0.20
CMTd-n151-k14	1162.55	1165.10	1164.12	0.98	0.22	0.14
CMTd-n200-k18	1395.85	1410.92	1406.46	4.46	1.08	0.76

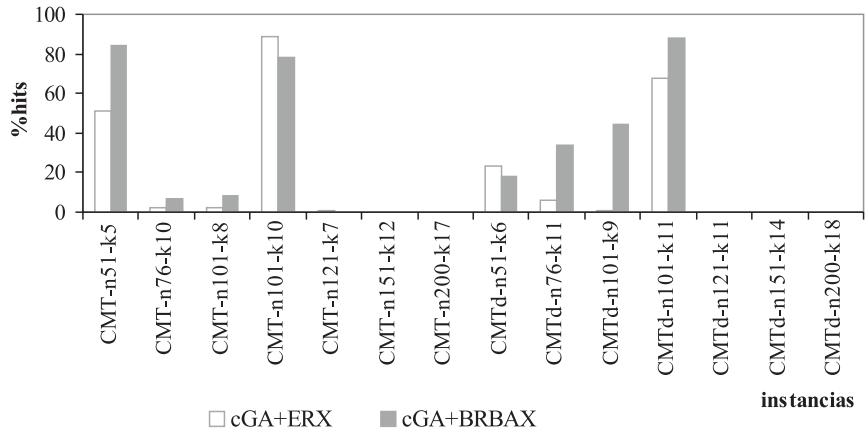


Figure 5: Comparison of the behavior of cGA+ERX and cGA+BRBAX in terms os success rate.

9 Conclusion

In this paper we have analyzed the behavior of improved GAs for solving the CVRP. Because an important factor in CVRP is the combination of customers to made a route, our GA is designed in such a way that offspring inherit grouping of customers. For this purpose, we have proposed a recombination operator (BRBAX) which generates offspring by combining appropriately the routes in both parents. We have compared this new problem specific operator with traditional ones present in the literature to solve permutation based problems. Moreover, we have analyze the impact of the incorporation of BRBAX in the behavior of a cellular GA. The study, validated from a statistical point of view, analyzes the capacity of the new recombination operator to improve the solution quality.

Our results show that the use of operators incorporating specific knowledge from the problem works accurately. This operator is based on the concept of building blocks, here defined as a group of customers which defines a route in the phenotype. This marks the difference with some of the traditional operators which randomly select the set of customers to be interchanged.

For obtaining a better solution, a local optimization is introduced in the cellular genetic algorithm using BRBAX. Comparing the behavior of this algorithm with a one present in the literature, our proposal obtains good quality of solutions with higher levels of success.

As a future work, we propose the construction of parallel versions of the algorithms studied in this work in order to improve the computational times. Furthermore, we plan to adapt the toroidal grid that holds all the individuals in the cGA to guide the search in the exploration or exploitation of the solution space depending of the search progress.

Acknowledgements

This work has been funded by the Universidad Nacional de La Pampa and the ANPCYT in Argentina from which we received continuous support.

References

[1] E. Alba and B. Dorronsoro. The exploration/exploitation tradeoff in dynamic cellular evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 9(2):126–142, April 2005.

[2] E. Alba and B. Dorronsoro. Computing nine new best-so-far solutions for capacitated VRP with a cellular genetic algorithm. *Information Processing Letters*, 98(6):225–230, June 2006.

[3] Enrique Alba, Francisco Almeida, Maria J. Blesa, J. Cabeza, Carlos Cotta, M. Díaz, Isabel Dorta, Joaquim Gabarró, Coromoto León, J. Luna, Luz Marina Moreno, C. Pablos, Jordi Petit, Angélica Rojas, and Fatos Xhafa. Mallba: A library of skeletons for combinatorial optimisation (research note). In Burkhard Monien and Rainer Feldmann, editors, *Euro-Par*, volume 2400 of *Lecture Notes in Computer Science*, pages 927–932. Springer, 2002.

[4] Enrique Alba and Bernabé Dorronsoro. A hybrid cellular genetic algorithm for the capacitated vehicle routing problem. In Ajith Abraham, Crina Grosan, and Witold Pedrycz, editors, *Engineering Evolutionary Intelligent Systems*, volume 82 of *Studies in Computational Intelligence*, pages 379–422. Springer, 2008.

[5] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. Oxford Univ. Press, 1997.

[6] Barrie M. Baker and M. A. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers and Operations Research*, 30(5):787 – 800, 2003.

[7] John E. Bell and Patrick R. McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1):41–48, 2004.

[8] Jean Berger and Mohamed Barkaoui. A hybrid genetic algorithm for the vehicle routing problem with time windows. In *Advances in Artificial Intelligence. 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 44–51. Morgan Kaufmann, 1998.

[9] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. *Combinatorial Optimization*, pages 315–338, 1979.

[10] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

[11] D. Croes. A method to solving traveling salesman problem. *Operations Research*, 6:791–812, 1958.

[12] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[13] Tim Duncan. Experiments in the use of neighbourhood search techniques for vehicle routing. Technical report, University of Edinburgh, 1995.

[14] D. B. Fogel. An evolutionary approach to the travelling salesman problem. *Biological Cybernetics*, 60(2):139–144, 1988.

[15] Mario Giacobini, Enrique Alba, Andrea Tettamanzi, and Marco Tomassini. *Modeling Selection Intensity for Toroidal Cellular Evolutionary Algorithms*, volume 3102 of *Lecture Notes in Computer Science*, pages 1138–1149. Springer, 2004.

[16] Mario Giacobini, Enrique Alba, and Marco Tomassini. Selection intensity in asynchronous cellular evolutionary algorithms. In Erick Cantú-Paz, James A. Foster, Kalyanmoy Deb, Lawrence Davis, Rajkumar Roy, Una-May O'Reilly, Hans-Georg Beyer, Russell K. Standish, Graham Kendall, Stewart W. Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, Mitchell A. Potter, Alan C. Schultz, Kathryn A. Dowsland, Natasa Jonoska, and Julian F. Miller, editors, *GECCO*, volume 2723 of *Lecture Notes in Computer Science*, pages 955–966. Springer, 2003.

[17] David E. Goldberg and Robert Lingle Jr. Alleles, loci, and the traveling salesman problem. In John J. Grefenstette, editor, *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, pages 154–159, Pittsburgh, PA, July 1985. Lawrence Erlbaum Associates.

[18] B.L. Golden, E.A. Wasil, J.P. Kelly, and I.M. Chao. *Metaheuristics in vehicle routing*, volume Fleet Management and Logistics, pages 33–56. Kiuwer, Boston, t.g crainic and c. laporte edition, 1998.

[19] Lenstra J. and Kan A.R. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227, 1981.

[20] Jcell. Software NEO group. University of Malaga.

[21] K.L. Mak and Z.G. Guo. A genetic algorithm for vehicle routing problems with stochastic demand and soft time windows. In *Systems and Information Engineering Design Symposium, 2004. Proceedings of the 2004 IEEE*, pages 183–190, April 2004.

[22] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolutions Programs (Third, Revised and Extended Edition)*. Springer-Verlag, Berlin Heidelberg New York, 1999.

[23] Yuichi Nagata and Olli Bräysy. Efficient local search limitation strategies for vehicle routing problems. In *EvoCOP*, pages 48–60, 2008.

[24] I.H Osman. Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem. *Annals of Operations Research*, 41, 1993.

[25] T. Potter and T. Bossomaier. Solving vehicle routing problems with genetic algorithms. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 2, pages 788–793 vol.2, Nov-1 Dec 1995.

[26] J. Sarma and K.A. De Jong. An analysis of local selection algorithms in a spatially structured evolutionary algorithm. In T. Bäck, editor, *ICGA97*, pages 181–186. Morgan Kaufmann, 1997.

[27] D. Skrlec, M. Filipec, and S. Krajcar. A heuristic modification of genetic algorithm used for solving the single depot capacited vehicle routing problem. In *Intelligent Information Systems, 1997. IIS '97. Proceedings*, pages 184–188, 8-10 1997.

[28] The VRP Web. http://neo.lcc.uma.es/radi-aeb/WebVRP/.

[29] Darell Whitley, Timothy Starkweather, and D'Ann Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. pages 133–140, 1989.

[30] Yi-Liang Xu, Mene-Hiot Lim, and Meng-Joo Er. Investigation on genetic representations for vehicle routing problem. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 4, pages 3083–3088 Vol. 4, Oct. 2005.